BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF INFORMATION SYSTEMS
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

VIDEO QUALITY MONITORING USING NETFLOW

SLEDOVÁNÍ KVALITY VIDEO PŘENOSŮ POMOCÍ TECHNOLOGIE NETFLOW

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR **JAN HAVLÍK**
AUTOR PRÁCE

SUPERVISOR **Ing. PETR MATOUŠEK, Ph.D., M.A.**
VEDOUCÍ PRÁCE

BRNO 2017

Zadání bakalářské práce/19961/2016/xhavli43

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2016/2017

Zadání bakalářské práce

Řešitel: **Havlík Jan**
Obor: Informační technologie
Téma: **Sledování kvality video přenosů pomocí technologie NetFlow**
Video Quality Monitoring Using NetFlow
Kategorie: Počítačové sítě

Pokyny:

1. Seznamte s monitorováním toků pomocí NetFlow. Využijte volně dostupné nástroje, např. nprobe a nfdump.
2. Navrhněte a implementujte nástroj pro detekci video přenosů ze síťových dat se zaměřením na typ přenášeného videa.
3. Do nástrojte implementujte algoritmus pro výpočet kvalitativních parametrů přenosu (např. typ kódu, ztrátovost paketů, rozptyl, zpoždění).
4. Integrujte vytvořený nástroj do NetFlow sondy. Navrhněte rozšíření NetFlow záznamů (verze 9 či IPFIX), které by přenášelo vypočítané kvalitativní parametry.
5. Ověřte chování aplikace v laboratoři, případně v produkční síti. Zhodnoťte výsledky a praktické použití.

Literatura:

- L. Sun, I. Mkwawa, E. Jammeh, E. Ifeachor: Guide to Voice and Video over IP. For Fixed and Mobile Networks. Springer, 2013.
- W. Simpson: Voice over IP: A Complete Guide to Understanding the Technology, Focal Press, 2013.
- B. Claise: Cisco Systems NetFlow Services Export Version 9, IETF RFC 3954, 2004
- Manuálové stránky nfdump.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Matoušek Petr, Ing., Ph.D., M.A.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 17. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstract

This bachelor's thesis is about creating new tools for a video transfer quality monitoring. It consists of a client-server architecture, where the client is gathering video quality statistics and passes those statistics to the server. The server updates relevant NetFlow IPFIX records with these statistics. The project includes video encoding, packet encapsulation and Internet protocols related to this topic. The architecture is written in a C language for a UNIX operating system.

Abstrakt

Tato bakalářská práce vznikla za účelem vytvoření nástrojů pro sledování kvality video přenosu na Internetu. Cílem je vytvořit takové nástroje, které automaticky rozpoznají přenos videa na Internetu z předloženého souboru v daném formátu a následně provedou jeho analýzu. Vybrané statistiky o video kvalitě posílá nástroj pomocí architektury klient-server na kolektor, kde server statistiky zpracuje a rozšíří relevantní Netflow IPFIX záznamy. Pojednává tedy i o problematice kódování videa, zapouzdření paketů a internetových protokolech souvisejících s daným tématem. Systém je psaný v jazyce C pro unixový operační systém.

Keywords

Video quality, NetFlow, IPFIX, Codec

Klíčová slova

Video kvalita, NetFlow, IPFIX, Kodek

Reference

HAVLÍK, Jan. *Video Quality Monitoring Using NetFlow*. Brno, 2017. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Petr Matoušek, Ph.D., M.A.

Video Quality Monitoring Using NetFlow

Declaration

Hereby I declare that this bachelor's thesis was prepared as an original author's work under the supervision of Ing. Petr Matoušek Ph. D., M.A. All the relevant information sources, which were used during preparation of this thesis, are properly cited and included in the list of references.

.....
Jan Havlík
May 17, 2017

Acknowledgements

I would like to take this opportunity to thank my supervisor Ing. Petr Matoušek Ph. D., M.A. for his valuable advice and helpful consultations. I would also like to thank my parents for their patience, my girlfriend Johanna for a moral support and to my friend Dan for his objective critical observations.

Contents

Chapter 1

Introduction

Motivation

The purpose of this work is to create a communicating system of multiple modules, consisting of a video quality monitoring tool and the tool for updating NetFlow records on a collector. Video quality monitoring tool will detect video packets, dissect those packets and analyze them for the purpose of getting an information, which can be used for a objective video quality measurement. Next part is a NetFlow updater in a form of a server, which will update relevant NetFlow records with the video quality statistics. [4]

Objective as an adverb is defined as follows:

> *Not influenced by personal feelings, interpretations, or prejudice; based on facts[7]*

In our case, the facts in the definition are the data from a dissected packet. We will use passive monitoring using a *NetFlow* probe on an exporter [4], which will collect the flow statistics on a collector.

Using these technologies, we can filter out and analyze packets for video quality monitoring, which is a necessity considering today's Internet usage. Because video transfer quality is dependent on an Internet service provider (ISP) and the route, which is chosen for a packet, we will monitor a video quality of a video packet flow on a route from a server (Youtube, PrimaPlay, iVysílání České televize, SIP User Agent Server) to the client (Personal computer). Or between two clients using a VoIP telephony. This set of tools can be used as an objective feedback on the video quality analyzed either from a capture file, or from a live capture, with the option to write those statistics to the NetFlow files.

The architecture of communicating modules is written in the C language and is intented to use for a UNIX platform.

Chapter 2

Video transmission over the Internet

2.1 Introduction

In order to measure the quality of a video transfer, we have to filter out packets containing video data. Because of the encapsulation of a packet, we have to dissect each packet according to the layers and filter out those we are interested in. We will use the TCP/IP model (2.2.1) for a network architecture to describe the process of a packet filtering.

2.2 Network architecture

This section describes the network architecture model we used and a brief description of the encapsulation concept to give a more detailed explanation of the packet dissection and filtering.

2.2.1 TCP/IP model

This model separates network into four layers as follows [30]:

- Network interface layer
- Internet layer
- Transport layer
- Application layer

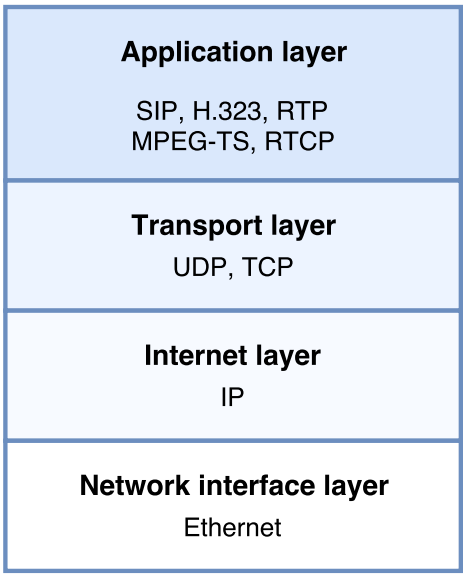

Figure 2.1: Each protocol operates on a different layer.

This figure also describes protocols used for a video transmission. Lower layer protocols will be mainly used for a packet filtering (data flow detection) and protocols from upper layer will be used for an objective video quality measurement.

2.3 Network Interface Layer

This section covers protocols from the lowest layer in the TCP/IP model in order to describe the video packet filtering process. The application layer will be split into subsections focusing on the signalling and transport protocols.

Network interface layer describes standards for physical media and electric signals [30]. There is only one protocol which I will focus on - Ethernet.

2.3.1 Ethernet frame

Ethernet frame is described in IEEE 802.3 [18]. Each ethernet frame starts with ethernet header, which consists of the following parts:

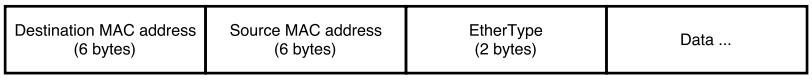

Figure 2.2: Ethernet II header according to IEEE 802.3-2015 specification

MAC address [9] is a unique Ethernet interface identifier.
EtherType [18] is a number of the protocol encapsulated in the frame payload.

The first packet filtering starts with an *EtherType*, where we look for an Internet Protocol (2.4) identification.

2.4 Internet Layer

Internet layer creates datagrams and ensures addressing and routing of those datagrams. This layer tries to find an optimal path for a datagram delivery (best-effort delivery) [30]. The most important protocol on this layer for us is an Internet protocol.

Internet protocol (IP)

IP protocol delivers packets from the source to the destination using an *IP address* [21]. There are two versions of an IP address.

- IPv4 - Internet Protocol version 4 (32 bits) described in RFC 791 [21]
- IPv6 - Internet Protocol version 6 (128 bits) described in RFC 2460 [6]

The need of the IPv6 implementation comes from the *IPv4 address exhaustion*, which happened in 2011 [16]. We can distinguish the IP version from the *EtherType* on a Network interface layer 2.3.1 as follows:

- IPv4 - 0x0800 [17]
- IPv6 - 0x86DD [17]

According to the values in the Ethernet header 2.3.1, we can throw away packets with different protocol on the Internet layer immediately.

2.5 Transport Layer

Transport layer creates a *logical connection* between the source computer application process and the destination computer application process. Transport protocols use a segmentation of the application data into the transport units called *packets* [30].

There are two main transport protocols which are important for the video transfer. Transmission Control Protocol (TCP [5]) and User Datagram Protocol (UDP [39]).

2.6 Application Layer

Application Layer is a layer created by processes and applications, which communicate with each other on the network. It is the highest layer of the TCP/IP model. The following list contains protocols important for the video transfer and the video quality measurement.

2.6.1 Signalling and controlling protocols

Signalling protocols are often used in the VoIP communication for the identification of the state of connection.

Session Initiation Protocol (SIP)

This signalling protocol is used for creating, modifying and terminating sessions with one or more participants (telephone calls, conferences, etc.). It enables Internet endpoints (user agents) to discover one another and agree on a session. This protocol also enables creation of an infrastructure of network hosts (proxy servers), for sending invitations and other requests [41].
SIP implements a three-way handshake [20].

- The caller sends an INVITE request
- The called side sends a 200 OK response to accept the call
- The caller sends an ACK response to indicate that the handshake is done and a video call is going to be setup[41]

Session Initiation Protocol uses the *port number 5060* send typically over the UDP or TCP transportation protocol.

Session Description Protocol (SDP)

SDP is a protocol for a session description with media details, transport addresses and metadata for participants. It does not work as a transport protocol, which is why it makes use of other protocols for this purpose (RTP, HTTP, etc.) [15]. This protocol is stored in the *Message Body* of a Session Initiation Protocol. This protocol contains the *Media Description*. *Media Description* consists of three important things for a video detection and video quality measurement as follows:

- *Media Type* - We focus on the video Media Type
- *Media Port* - Provides us with the information which RTP port contains video packets
- *Media Description* - Contains information about the coding algorithm

After obtaining this information, we can store the flow as a flow containing video packets and start measuring the video quality metrics for these packets [15].

Real-Time Streaming protocol (RTSP)

This protocol is used to establish and control media session between a client and a server. RTSP, unlike HTTP, does not treat request as an independent transaction unrelated to previous requests. RTSP is not used for data transmission. RTP and RTCP protocols are used for this purpose [44].

Real-Time Streaming protocol provides us with the real-time control of the video transfer.

2.6.2 Transport protocols

Transport protocols on the Application Layer often add more useful information about the video transfer (RTCP, RTP) or create a digital container format for transmission and storage (MPEG-TS). We will use the information from these protocols to measure the video quality or filter out the video transfer.

Real-time transport protocol (RTP)

This protocol supports these video codecs useful for us [42]:

- H263 (2.7.1)
- H263-1998 (H263+ 2.7.2)
- MPEG-2 (2.7.3)
- MPEG-4 (2.7.4)

RTP specification describes RTP and RTCP sub-protocols. [43]

- RTP: Contains the information about timestamps, sequence number and payload format.
- RTCP: Contains the information about SSRC (Multiple Synchronization Sources).

If a video image is present in multiple packets, they share the same timestamp.

MPEG Transport stream (MPEG-TS)

This is a standard protocol for a video transmission. The encapsulation is designed for a less reliable transmission and services based on MPEG-2. The elementary stream data from a video encoder often create Packetized Elementary Stream (PES) packets and these packets are encapsulated inside the TS packets. This protocol is using tiny packet size (at most 188-byte sections), which leads to a great error resilience important for video conferencing. Each packet consists of the 4-byte header, which contains a *Sync byte* with the value of 0x47 [12]. This value will be used for a video packet filtering via the MPEG-TS detection. The important value is the *Continuity Counter (CC)*, which is a 4 bit number describing the packet arrival order. At first, we have to look into the Program Association Table (PAT) and find the information about the Program ID (PID), which is currently active. Then, we have to look into the Program Map Table (PMT), which consists of all the current streams. Each stream has its own Program ID. The content type of the stream is identified by an 8-bit descriptor tag.

Dynamic Adaptive Bitrate Streaming (DASH)

This adaptive bitrate streaming technique enables high quality streaming over HTTP. It detects the user's bandwidth and CPU capacity in real time to adjust the quality of a video stream accordingly. It uses a TCP transport protocol. The multimedia file is split into partitions and delivered to the client by HTTP. The specification provides formats for use with MPEG-2 and MPEG-4 [22].

2.7 Video codecs

A *codec* is a coding algorithm, which is used for the recoding of decimal values to a binary form. A codec's main purpose is a reduction of the transferred data volume. Video codec knowledge for a given video stream is a very important information for us because of the volume reduction possibilities. The reduction efficiency can even be twice that high compared to the previous version of a coding algorithm.

2.7.1 H263

Successor of the H261 [10] codec, which contains half of its bitrates. This codec was subsequently refined to the H263+ (H263-1998) codec. Normally, it is used for VoIP and 3G mobile handsets for video call applications.
A H263 codec has seven picture types, but only two are mandatory [53]:

- I-picture: Intra-coded picture with no reference to other pictures for prediction.
- P-picture: Inter-coded picture, with the use of previous pictures for prediction, which removes temporal redundancy.

Each H263 picture consists of a Group of Blocks (GoB) and each GoB contains one row of macroblocks [47]. Every macroblock contains four blocks of luminance components and two blocks of chrominance components as shown in 2.3.

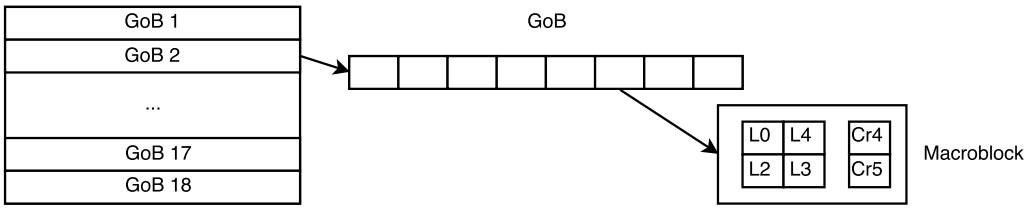

Figure 2.3: H263 picture structure

2.7.2 H263-1998 (H263+)

It is a more common payload format than H263 and it is recommended for new implementations. This codec adds numerous coding options to improve codec performance over its predecessor, H263.

New coding options are as follows [2]:

- Slice structured mode: Minimizes video delay and provides advanced error resilience capability.
- Independent segment decoding (ISD): Each video picture frame is broken into segments and encoded separately.
- Allows use of an older reference picture rather than the one preceding the current picture.

2.7.3 MPEG-2

This coding standard is also known as H262 and was developed for Digital Video Broadcasting (DVB). It is used for a high quality video with higher bitrates, not suitable for low bit rate applications such as VoIP applications. Either past or future picture can be used as a reference picture and the reference picture can be located more than one picture away from the current encoded picture [47] [34].

2.7.4 MPEG-4

The MPEG-4 coding standard has many similarities to the H263 design. It is also similar to the MPEG-2, but it is much more complex. This coding standard has the ability to code multiple objects within a video frame. The main purpose of MPEG-4 is storing and delivering multimedia content over the Internet. It has many application profiles and the simple profile (level 0) is used in 3G video call applications. It has a high coding efficiency and a high error resiliency. [47]

2.8 Type of service

2.8.1 Video on Demand (VoD)

Video on demand is a model for video viewing on the client side. Client requests a video content from the server and the server starts transmitting video data back to the client. Examples of this type of services are YouTube [19] or local televisions like Česká televize (iVysílání České televize [49]) and Prima (Prima Play [14]). This service uses a HTTP protocol [13].

2.8.2 Voice over IP (VoIP)

Voice over IP is a service for making telephone calls similar to classic telephone network - PTSN (Public Switched Telephone Network). In this work, we focus on a VoIP communication with the use of the SIP/SDP signalling protocols (2.6.1) and RTP transport protocols (2.6.2).

2.8.3 Video streaming

Video streaming is a service using the *multicast* communication in order to send the video data to multiple subscribed clients at once. We focus on the MPEG Transport Stream video streaming described in 2.6.2.

2.9 NetFlow v9

NetFlow version 9 Flow-Record format is a standard provided by Cisco IOS®. This format provides access to the IP flows data for network administrators. Network administrators can use this information for network management, planning, Internet Service Provider billing or defending against a Distributed Denial of Service (DoS) attacks. The output of NetFlow is a *flow record*. This standard is adjustable, which means that it can provide support for the future protocols. [4]

Every NetFlow Version 9 record format contains a packet header followed by Template or Data FlowSets. Template FlowSet provides description for Data FlowSets that follow. The figure below describes the NetFlow Version 9 record format:

Figure 2.4: NetFlow Version 9 Export Packet [4]

2.10 NetFlow IPFIX

NetFlow IP Flow Information Export (IPFIX) protocol originated from the NetFlow version 9 protocol by Cisco IOS®. It is similar to the previous standard and also provides network administrators with access to the IP Flow information. However, the IP Flow Information Export protocol provides more improvements than the NetFlow version 9. NetFlow IPFIX brings us the optimizations like a method for exporting complex data structures [RFC6313] or the bandwidth-saving method for the IPFIX protocol [RFC5473]. Here is a description of the main differences compared to the NetFlow version 9[3]:

- Slightly different terminology (e.g.: Field ID1 `IN_BYTES` versus `octetDeltaCount`)
- Variable length fields in IP Flow Information Export
- 238 field types against the 79 defined in NetFlow v9 [37]

In this work, the most important difference is the last point of the list. Because we export the video quality statistics in a serialized format (5.2.1) as a string, we had to find a suitable field, which will contain such information. We decided for the *userName* field, which has a *String* data type. This field is useful for us, because the `libnf` library (5.1.2) also supports the *userName* field.

Chapter 3

Video quality measurement

The main purpose of our work is a video quality measurement from the data encapsulated in a packet. The motivation is the increase of the video data traffic on the Internet. A video can express more than a picture and a picture can express more than words. The basic example is YouTube service, where people can watch music video clips, instructional videos or news all over the world. Many people also use Video on Demand, Voice over IP or Video streaming services and for each of those services, we want to measure the transferred video quality.

3.1 Subjective video quality measurement

The subjective video quality measurement is a human rated video quality. It measures, how a human viewer rates given video sample and adjusts the quality representation based on human opinion. We mention this rating, because the existing tools for video quality measurement use the subjective approach to this problematic. Subjective quality assessment methods for multimedia applications are described in the ITU-T P.910 recommendation [26].

3.1.1 Absolute Category Rating (ACR)

Absolute Category Rating is a subjective quality rating method, where the test video sequences are presented one at a time. Each subject is asked to evaluate the quality of each sequence after watching it. The scale for rating is very simple five-level scale:

1. Bad
2. Poor
3. Fair
4. Good
5. Excellent

This scale might be extended if needed, to a nine-level scale. There are many variations of this method (e.g. ACR-HR, DCR, ...), but those often use a reference video for a quality rating. ACR is described in the ITU-T P.910 recommendation [26].

3.1.2 Mean Opinion Score (MOS)

Mean Opinion Score is widely used for telephony quality measurement. It is a subjective method and the conditions for quality measurement are defined in ITU-T P.10/G.100 recommendation [25].

This recommendation specifies many aspects for testing environment, including speech material (it should be a maximum of five short, meaningful sentences, for example "You will have to be very quiet.").

The specifications for a testing room are as follows:

- A quiet room
- Volume between 30 and 120 m^3
- Reverberation time less than 500 ms
- Room noise level below 30 dBA

MOS is defined for audio only with many variations, which may differ in objectivity/subjectivity, listening-only/talking-only, noise level or speech quality [25]. We could not find a suitable video quality MOS value replacment, which would include all of the video quality network characteristics (3.2.2).

3.2 Objective video quality measurement

Objective video quality measurement focuses on mathematical models, which calculate the video quality from the video data. Objective video quality measurement does not involve the human perception, but the goal is to get the approximate results from a subjective video quality measurement.

Objective video quality metrics can be divided into three categories as follows:

1. *Full Reference Methods (FR)*
   Full reference methods use reference video and compare these two video samples to compute the video quality. These methods use signal or pixel comparison and do not include transmission or encoding process in between. We will not use this method for our quality measurement, but we can use some of the tools to validate our results [24].

2. *No-reference methods (NR)*
   No-reference methods compute the video quality without any knowledge about the source (reference) signal. Our tool is using the *No-reference objective video quality assessment*.

3. *Reduced Reference Methods (RR)*
   Reduced Reference methods combine both of the Full Reference and No-reference methods together [23].

The following characteristics are used for No-reference video quality assessment.
We can approach the quality assessment from the *Image characteristics* or from the *Network characteristics*.

3.2.1 Image characteristics

Quality assessment from the image characteristics often uses *Full reference methods* (1). Thus, we need a reference video to compute the video quality loss in the process of encoding, decoding and transferring via packets. These characteristics use a signal or pixel comparison to calculate the quality difference between the reference video and the evaluated video.

I will describe the methods for a video quality measurement which are most popular and most efficient, given by the full reference video quality assessment tools (4). First two basic and widely used methods are *Mean squared error* and *Peak signal-to-noise ratio*.

Mean squared error (MSE)

Mean squared error is a statistical value, which measures the average of the squares of the errors between the reference video and the evaluated video. MSE is calculated as follows:

$$MSE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I_{(i,j)} - K_{(i,j)}]^2, \tag{3.1}$$

where I is a frame from the reference video and J is a frame from the evaluated (distorted) video. Values m,n are pixel positions [8].

Peak signal-to-noise ratio (PSNR)

Peak signal-to-noise ratio is an improved method for objective video quality assessment, which uses the MSE formula (3.1). It is the ratio between maximum possible signal in the video frame and the noise, which is corrupting the signal accuracy (which, in our case, is the *MSE value*). PSNR is calculated as follows [40]:

$$PSNR = 20 \cdot \log_{10}(MAX_I) - 10 \cdot \log_{10}(MSE), \tag{3.2}$$

where MAX is a maximum of a possible signal or pixel value (e.g. if the pixel is represented with 1 byte, the value is 255) and MSE is the Mean squared error value.

MSE and PSNR are *full reference objective metrics*. In order to approximate the human video quality perception, another method was invented, which correlates with the subjective video quality results - *Structural similarity*.

Structural similarity (SSIM)

The structural similarity is a full reference metric which improves the PSNR metric. SSIM uses knowledge about human visual perception and focuses on the perceived change in *structural information*. Structural information means, that the pixels close to each other are strongly dependant on each other. Structure similarity is calculated with the following formula [51]:

$$SSIM_{x,y} = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}, \tag{3.3}$$

where μ_x is the average of x, μ_y is the average of y, σ_x^2 is the variance of x, σ_y^2 is the average of y and σ_{xy} is the covariance of x *and* y.

$c_1 = (0.01 \cdot L)^2$ and
$c_2 = (0.03 \cdot L)^2$, where L is the range of pixel values.

This formula is often applied on the brightness of the image (*luma*), but it can also be applied on the RGB values of the selected frame. The result is a value within the interval $< -1; 1 >$, whereby the value of 1 is reachable only for two identical sets of data.

3.2.2 Network characteristics

Video quality assessment from the network characteristics is based on the *packet analysis*. The video packet analysis contains the no-reference methods for a video quality assessment. In our case, we use selected information from the whole packet, which means that we use the *Deep packet inspection (DPI)* [38] for this purpose.

In general, the video data corruption during the network transfer leads to a visual imperfection of the transferred video file. We use the following methods with a description for the video quality assessment:

One-Way Packet Loss

One-way packet loss is a number of packets lost over a pre-defined measurement period. The measurement unit is a integer number of lost packets [1].
The metric depends on the used transport protocol. In case of the TCP protocol ([5]), we can calculate One-way packet loss on the transport layer. For the UDP protocol, we have to dig into the application layer.

1. **Transport layer**

   - *TCP*
     Transmission control protocol implements a *sequence number* generated as a process of the *three-way handshake* We can detect a missing packet from *monitoring these sequence numbers and comparing the values from previous and next accepted packet*. The absolute sequence number is a random number from the interval $< 0; 2^{16} - 1 >$ [46]. We also mention the relative sequence number, which is often used in the network analyzers like *Wireshark*. The relative number is a number relative to the packet flow in a capture file to increase the readability and orientation in the file [31].

2. **Application layer**

   - *RTP for the UDP transport protocol*
     User datagram protocol has no sequence numbers implemented by the specification ([39]). That is why we use the RTP from the application layer to control the packet sequence. Real-time transport protocol incorporates a *sequence number* [43]. We can detect the out-of-order packets by comparing the sequence number from the previous and current packet.

   - *MPEG-TS for the UDP transport protocol*
     Each MPEG-TS elementary stream contains information about the *continuity counter* (2.6.2). If the *continuity counter* between the current and previous MPEG-TS *elementary stream* packet differs, this packet is out-of-order.

Media Loss Rate

Media Loss Rate is closely related to One-Way Packet Loss. This metric is more specific than One-Way Packet Loss and is defined as *number of packets lost per one second*. We need this metric to monitor quality alteration in real time, while monitoring the network and analyzing any video streams. The measurement unit is integer number of packets lost per one second.

We have to reset the time counter for every stream detected on our probe while analyzing the network. Then, we divide One-Way Packet Loss value with the value in the time counter. This leads to a multi-process application.
Common acceptable average MLR for Video on Demand in the *Service Level Agreement* is 0.004. [48]

Delay

Delay is a time difference between two subsequent packets. In our case, we have to use an approximated value, because we do not have the data to calculate a round trip delay, which is more precise. Round trip delay is a delay, which occurs when we send a packet and we accept the ACK after receiving the packet on the receiver's side. This is not possible to measure, because we usually analyze packet flow only in one direction. The approximation is made as follows:[29]

$$D_{i,j} = (TR_j - TR_i) - (TS_j - TS_i), \tag{3.4}$$

where TR_i is a time of arrival of packet i, TR_j is a time of arrival of packet j, TS_j is a timestamp of the packet j and TS_i is a timestamp of the packet i. The values from the formula are depicted on the following figure:

Timestamp is added only if we have the information from the protocol, e.g. timestamp in the RTP header (2.6.2). If the timestamp is present, we have to divide the value by the sampling frequency for the given codec.

The *sampling frequency* is the same value of *90 000 Hz*[42] for the following video codecs:

- H261

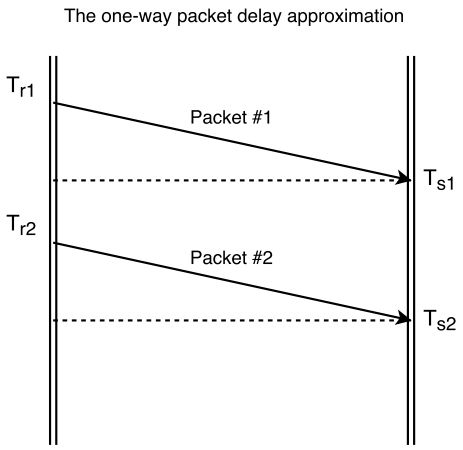

Figure 3.1:

- H263
- H263-1998
- H264
- MP2T
- mpeg4-generic
- MPV

The formula for the average one-way delay is as follows:[29]

$$d_1 = 0, \tag{3.5}$$

$$d_i = \frac{(d_{i-1} \cdot (i-1)) + D_{i,j}}{i}, \tag{3.6}$$

where d_1 is the first delay, set to zero, d_i is the delay for a i-th packet, d_{i-1} is the delay from the previous packet and $D_{i,j}$ is the approximated value calculated according to the formula 3.4.

Cumulative inter-arrival jitter

Jitter is a deviation from the packet arrival values. For our jitter formula, we use the value of a time difference $D_{i,j}$(3.4) as follows:[29]

$$J_1 = 0 \tag{3.7}$$

$$J_i = J_{i-1} + \frac{|D_{i-1,i}| - J_{i-1}}{16} \tag{3.8}$$

Media Delivery Index (MDI)

Media delivery index is a proposed metric for a accurate jitter and delay measurement at the network level. Packet delay variation and a packet loss (due to the buffer overflow) are key characteristics in video quality measurement. MDI consists of two components, *Delay Factor* and *Media Loss Rate*. The latter was previously described in 3.2.2. Delay Factor is a value, that indicates how much milliseconds does it take to drain the virtual buffer, which is used to eliminate jitter. *Maximum acceptable Delay Factor is in the range of 9 - 50 ms* [27]. Media delivery index can be calculated as follows: [27]

$$DF : MLR \tag{3.9}$$

$$DF = \frac{VB(max) - VB(min)}{MediaRate}, \tag{3.10}$$

where VB is a Virtual buffer size calculated as follows:

$$VB = ReceivedBytes - DrainedBytes \tag{3.11}$$

and last, but not least, the *Media Loss Rate*:

$$MLR = \frac{OutOfOrderPackets}{Time} \tag{3.12}$$

Media Bitrate

Media bitrate is a number of bits transferred or processed per unit of time. For the probe implementation, we set the unit of time as one second.
Thus the measurement unit is number of bits per second (bps, bit/s). The bitrate affects transferred video quality, but we have to consider coding bitrate for different codecs. We will use this metric only if we know the video codec. The formula for the media bitrate calculation is as follows:

$$R = \frac{P_s \cdot 8}{t}, \tag{3.13}$$

where R is the media bitrate in bps, P_s is the packet size and t is the time measurement interval for the selected flow.

Time-stamped Delay Factor (TS-DF)

At first, we have to define the original Delay Factor.

> *Delay Factor* - A temporal value given in milliseconds that indicates how much time is required to drain the virtual buffer at the concrete network node and at a specific time. In other words, it is a time value indicating how many milliseconds' worth of data the buffers must be able to contain in order to eliminate time distortions (jitter). [11]

This would be hard to implement to our work, because of the virtual buffer drain rate calculation, so we chose a different approach. In order to calculate the Media Delivery Index value (3.2.2) according to the *RFC4445* standard[27], we need to know the value

of a Delay Factor. The Time-stamped delay factor is a possible solution to this problem, because it uses a new method for the Delay factor calculation.

Time-stamped Delay Factor uses the Delay value implemented according to the *RFC3550*[43] mentioned in the part dealing with Delay (3.4). In order to calculate the Time-stamped Delay Factor, we have to keep the maximal and minimal value of a Delay over the time period. Then, at the end of that time period, we can calculate the Time-stamped Delay factor as follows:

$$\text{TS-DF} = D_{max} - D_{min},$$

where D_{max} is a maximal Delay value over the time period and D_{min} is a minimal Delay value over the time period.

Time-stamped delay Factor is not sensitive to the case of packet loss (which is part of the Media Delivery Index calculation), but simulations show, that it has comparable results to constant bitrate streams.[28]

Chapter 4

Related work

The related work describes similar tools for a video quality assessment. There are software and hardware solutions for video quality assessment, but *only for the reference methods*. The advantage of those tools is the objective and subjective video quality assessment, which is possible due to the reference methods.

4.1 MSU Video Quality Measurement Tool

This is the full-reference video quality measurement tool implemented by the *MSU Graphics and Media Lab (Video Group)* This tool uses the PSNR and SSIM metrics for the video comparison, supports wide range of video formats and codecs and uses multiple visualization techniques for the results. Between many contributors to this tool are companies like Cisco, Oracle, NASA, Sony and Skype. This program outputs results in a CSV or JSON file and automatically saves error video frames in a bmp file.[50]

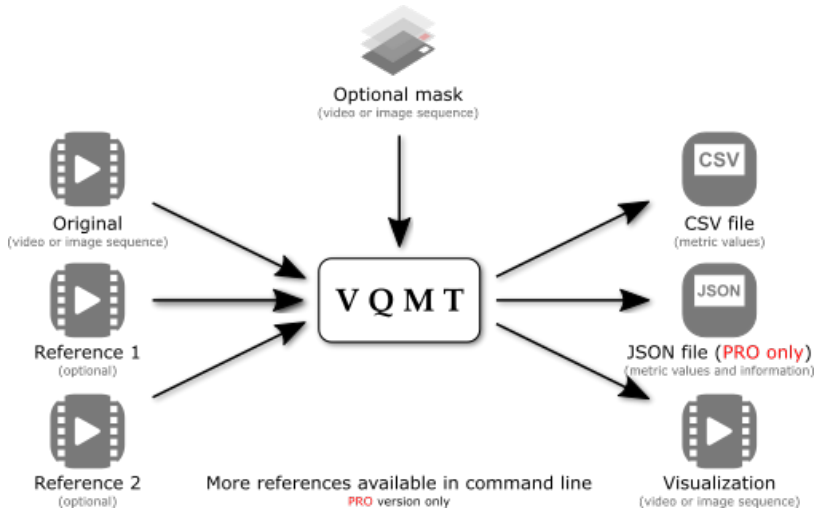

Figure 4.1: The video quality assessment process in VQMT

4.2 PEVQ – the Standard for Perceptual Evaluation of Video Quality

PEVQ uses full-reference metrics according to the Video Quality Experts Group [52] and the *J.247 ITU-T* recommendation [24]. These are some of the output metrics from this tool:

- PEVQ MOS - value according to the J.247 recommendation
- PSNR
- Delay
- Brightness
- Contrast

The principle of this tool is described on the following figure: [36]

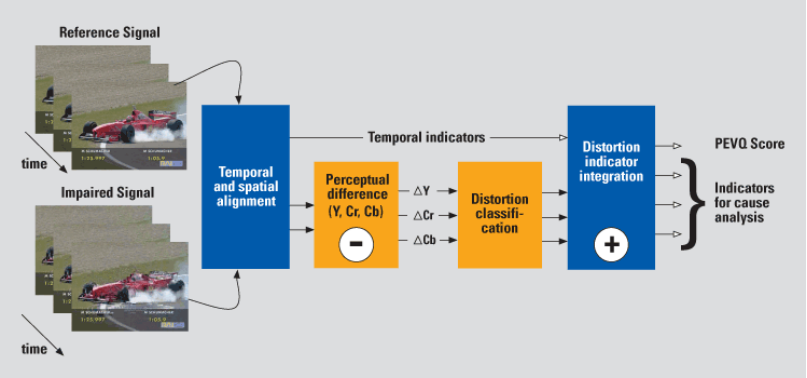

Figure 4.2: PEVQ principle

4.3 ProLab - Audio and Video Quality Testing Solution

This is a commerce tool for real-time and passive off-line measurement and analysis. The video quality monitor implemented by this company uses *no-reference subjective and objective metrics*. The result MOS value is dependant on a video codec and corresponds with the human perception of a video quality [45].

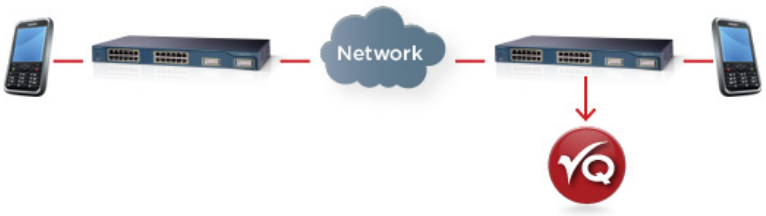

Figure 4.3: The architecture of the VQMonitor tool.

4.4 Video Quality Monitor

Video Quality Monitor flexible no-reference video quality measurement tool, which monitors the Quality of Experience [33]. It can measure quality in off-line mode with a reference file or you can start a on-line capture mode for the streamed video quality measurement. This tool supports multithreading and implements their own web server to access and analyzes the results from multiple points. The result is a MOS value of the analyzes video stream or file, blur, contrast and jerkiness.

	VQMT (MSU)	**PEVQ (Opticom)**	**ProLab (Spirent)**	**VQM (AccepTV)**
Operating system	Windows	Windows, Linux, Android	Any	Windows
License	PRO version, Free version with a few features also available	Commercial tool	Commercial tool	Commercial tool
Reference metric	Full reference	Full reference	No reference, Full reference	No reference, Hybrid
Input	Original File, Compressed file, Metric selection (VQM, SSIM, PSNR, MSU, MSE)	Video test sequences (Reference and test file)	Wireshark capture files or 3GP file formats	Capture file, Video file
Output	Original File, Compressed file, Metric selection (VQM, SSIM, PSNR, MSU, MSE)	MOS	MOS, Packet Size, Bandwidth, Packet Loss, Duplicate packets, frame rate, Jitter	MOS, bitrate
Disadvantages	Full reference metric, which cannot be objectively compared to out tools	Full reference metric, which cannot be objectively compared to out tools	Not possible to get a student license, because of the complex hardware solution. Could not be tested	Produces incorrect MOS values, similar to R-factor scale, but the lowest values of the scale (10-50)

Table 4.1: Video quality measurement tools comparison

Chapter 5

Implementation

The video quality measurement tool is written in the C language. We chose this language because of the processing speed needed for the packet analysis. The first implementation was written in the Python language, but the interpreter was too slow to analyze a packet flow in real time.

5.1 Libraries and tools

In this section, I will describe the modules and libraries, which I used for the tool. I will not mention the C standard libraries, but I will focus on the external libraries imported into the project.

5.1.1 Libpcap 1.8.1

This library provides us with the constants and functions needed for the packet capture and sniffing. At first, we have to set the network card to *promiscuous mode*, which will ensure that all of the network traffic passing through will be taken into consideration. Then, we can either use *live capture* or the *pcap file analysis*. For the live capture, we have to set the device we want to listen on first (e.g.: *eth0, x11*, etc.).

Live capture uses the function `pcap_open_live()` and starts listening on the selected device. This function also uses *timeout*, which is useful for our application. If there is no video data detection for a certain period of time, the program shuts itself. Secondary option is to use the function `pcap_open_offline()`, which opens a pcap file and returns handle to this file.

After setting the environment like this, we can pass the parameters through our packet handling function. Note that for compiling the *libpcap* library with our project, we have to add the `-lpcap` directive to the Makefile.

We implemented Libpcap on the client side of the tool, where it can analyse packets flowing through the client machine and then send the video quality statistics to the collector. The `libpcap` library can be installed with the following command (for the Ubuntu linux distribution):

```
sudo apt-get install libpcap-dev
```

5.1.2 Libnf 1.25

Libnf is a library for manipulating with NetFlow records and files (2.9). It is a library created by Ing. Tomáš Podermański from the Brno university of Technology. Libnf provides the access to the NetFlow record files. At first we have to open desired NetFlow record file. Then, we can filter the flows in the NetFlow record file and set the value of the NetFlow field to our value.

In order to add a NetFlow record to the file, we have to open an input and an output file, otherwise it would get overwritten. Afterwards, we can delete the original file and rename the updated file.

We implemented Libnf on the server side of the tool, where it runs together with the collector and updates NetFlow records with the video quality statistics in serialized format sent by the client machine.

5.1.3 Nfcapd

Nfcapd is a NetFlow capture daemon, which accepts the NetFlow data and converts them into the NetFlow record files. Each file has the following format:

nfcapd.YYYYMMDDhhmm

For example the file *nfcapd.201704031337* stores the NetFlow records from May the 4th, 13:37 in the year 2017 onward.

Nfcapd is a part of the *nfdump* tools, but in order to correctly create NetFlow record, which we can write to and read from, we have to compile this tool with a *NSEL/ASA support*. This is because of the fact that we store our video quality statistics into a char array, with the size of 256 characters. This is the size of a libnf constant `LNF_STRING` used for the *username* field in libnf library and *username* field is supported by nfcapd and nfdump only with the *NSEL/ASA support*. The reason to use this field is described in subsection 5.2.1.

In our work, the *nfcapd* daemon will work as a collector on the server side and it will generate the NetFlow record files. We can set up the intensity of a NetFlow record creation and specify the port and address this tool will be listening on.

5.1.4 Softflowd

Softflowd is a NetFlow traffic analyser, which can be used as an exporter for the NetFlow data. We used this tool for testing purposes, because of the absence of a router able to export NetFlow data. Softflowd can also replay and export information from the pcap files to the collector, which is useful for testing purposes. [32]

Together with the nfcapd collector, we can create NetFlow records from captured packets in pcap files as follows:

1. Run the capture daemon as a collector `nfcapd -p 12345 -l ./netflow_records`
2. Export the data from pcap file `softflowd -u 127.0.0.1:12345 -r ./test.pcap`

After a set period of time, there will be a NetFlow record stored in the `netflow_records` folder with data and flows from the input pcap file.

5.1.5 nProbe

The *nProbe* is a NetFlow v5/v9/IPFIX probe for IPv4/v6. *nProbe* can be either used as a NetFlow probe, a NetFlow collector or a NetFlow exporter. It is a tool developed by the *ntop* company [35]. This probe allows us to export the NetFlow IPFIX packets to a collector (`nfcapd`) and create relevant NetFlow records, when we do not have other options for a NetFlow packets export. This tool was mainly used for testing purposes, to gather the network traffic and export NetFlow packets to the collector.

5.2 Architecture of the tool

The tool consists of two parts. The main part is a client machine program *vidq*, which analyzes the captured packets and outputs video quality information in a serialized format. This output is sent to the server side of the tool, which consists of a *nfcapd* collector and a *vidq_updater*, which uses the libnf library to update the flows containing video data in the NetFlow records. The following figure describes the architecture and communication between the parts:

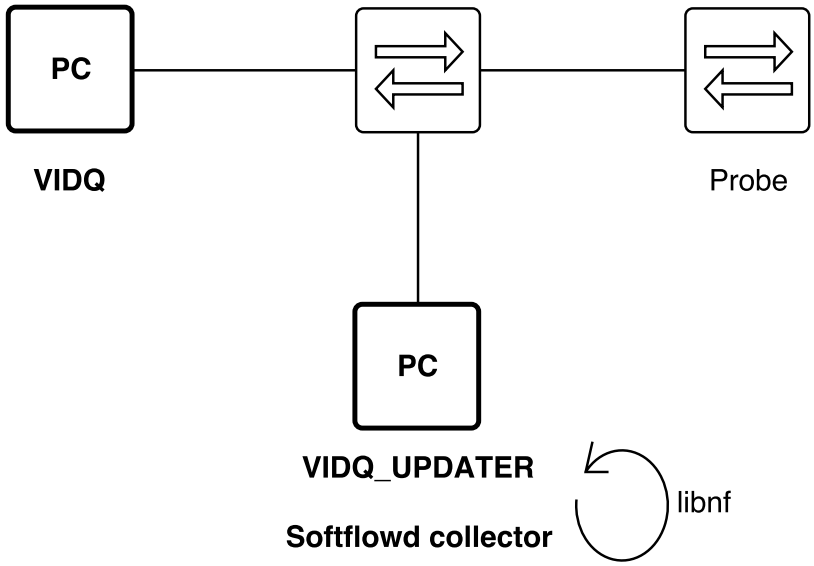

Figure 5.1: The architecture of the tool.

5.2.1 vidq

Vidq is the core part of the video quality measurement system. It allows the use of a online and offline mode and the purpose of the tool is to *detect, analyze and evaluate* the network flows containing video data. Vidq analysis produces the following information about the network flow containing video data:

- Video codec (if the information is present)
- Media bitrate in `kilobytes per second` (3.2.2)
- Cumulative Inter-Arrival Jitter in `miliseconds` (3.2.2)
- One-way Delay in `miliseconds` (3.4)

- Media Delivery Index (3.2.2), consisting of a Time-stamped Delay Factor (3.2.2) and Media Loss Rate (3.2.2)

All of those values are *serialized* together with a unique flow identifier, which consists of a destination IP address and a destination port number.

Serialization

Serialization is a process, in which we convert our video characteristics into a *string* (given the absence of strings in C language, the format is a LNF_STRING type from the libnf library, which is actually an array of 256 characters). Together with those statistics, we are sending the filter and options information for the server. The serialized string has the following format:

```
ip_address:port,ts=value;fmt=mode
btr=value;j=value;d=value;mdi=delay_factor:media_loss_rate
```

`ip_address` is a destination IP address,
`port` is a destination flow port,
`ts` is a timestamp from the last video flow packet,
`fmt` is a capture format,
`btr` is a media bitrate of the flow,
`j` is a cumulative inter-arrival jitter,
`d` is a one-way delay and
`mdi` is a media delivery index

The format field can contain following values:

- `l` - Passed serialized string is from a *live capture*
- `f` - Passed serialized string is from a *pcap file*
- `t` - Passed serialized string is from a *pcap file*, but we do not check the timestamp boundaries and check all the present NetFlow records in given folder

We can later split those values based on the equality sign and the semicolon.

Video data detection

In our work, we are able to detect three common types of video transfer:

- Video on Demand (2.8.1)

  Video on Demand uses the HTTP requests for a video data request and then proceeds to transport those packets via the TCP transport protocol. In the beginning, we detect the HTTP GET requests. The HTTP GET request contains information about the expected data, but not about the content of the flow. That is why we have to select only those requests, accepting all the types of flows, which is written as `Accept: */*`. We save the destination port number of such flow into an array and repeat the process for every new port number. Then, when we detect the HTTP 200 OK response, we look inside the packet and focus on the `Content-Type` field. If there is a *video* Content-Type, we add this information into the structure. After the end of the flow, or when we reach an end of the pcap file, we iterate through the array of

structures and remove flow records without the video content. We monitor the TCP packets and metrics we are interested in from those packets.

- Voice over IP (2.8.2)

  Voice over IP also uses a video transfer. This happens when there is a video conference or a video call between two clients. In the beginning, there is a signalling between two clients or between clients and a control panel. We focus on the SIP/SDP protocol (2.6.1). We detect the SIP INVITE request. In the body of this packet, there is a Session Description Protocol, which contains the *Media description* and *Media port* of a call. Media description contains information whether there are video packets or not. We add the port into the structure array and monitor the metrics we are interested in from the Real-Time transport protocol (2.6.2) on the application layer.

- MPEG Transport Stream (2.6.2)

  MPEG Transport Stream is often connected to the video multicast streaming. For the video quality detection, we have to find the Program Association Table, which contains the Program ID of the packet. This packet contains Program Map Table. Program Map Table consists of multiple *Stream types*. We are interested in the following Program Element Descriptor Tags:
  - 0x01, Video stream header parameters for MPEG-1
  - 0x02, Video stream header parameters for ITU-T Rec. H.262
  - 0x10, Video stream header parameters for ISO/IEC 14496-2 (MPEG-4 H.263 based)
  - 0x1B, Video stream header parameters for ITU-T Rec. H.264
  - 0x24, Video stream header parameters for ITU-T Rec. H.265

Dynamic array allocation

In order to continually increase and decrease number of records in an array of video flows, we had to implement the dynamic array allocation using the `malloc`, `calloc`, `realloc` and `free` functions. In the beginning, we allocate space for each type of the video flow. Increasing the size of the array of structures is implemented with the `realloc` function every time there is a new video port detected. After the live video flow ends or we reach the end of a pcap file, we will free every flow which does not contain any video data. This flow (not containing video data) is swapped with the last item in an array and freed with the `free` function. Slightly different approach is to select flows containing video data, store them into a new array and remove the whole array with non-video flows. This may be usable in the situation, where we detect only few video flows and multiple flows suspected from containing video data, but not actually containing them in the end.

5.2.2 vidq_updater

`Vidq_updater` is a server running on a different network node together with a `nfcapd` collector. The program accepts the serialized data format passed by the `vidq` client, which gives it an information for the NetFlow record file selection according to the last timestamp of the flow and for detecting the given flow in the NetFlow record file. If the searching pattern matches successfully, `vidq_updater` updates the selected flows in the files with the video quality detected for the given flow.

NetFlow record file selection

The server accepts the flow timestamp from the serialized string. Then, it compares the timestamp with the timestamp obtained from the `nfcapd` record file suffix. There is a range in which the timestamp from the flow will be accepted. After a successful comparison `vidq_updater` writes the statistics into the file.

Updating the flows

After selecting a suitable flow record file, `vidq_updater` updates the flows as follows:

1. Finds the record according to the information from the serialized string (destination ip address, destination port)
2. Selects the `username` field
3. Updates the field with the video quality metrics in serialized format

Note that these operations are implemented with the `libnf` library using standard functions. Because of the `username` field usage, we need the IPFIX (2.10) support to be implemented. The updated file can be viewed afterwards with the `nfdump` tool, but the tool has to be configured with the `-enable-nsel` option, because the `username` field is part of the *NSEL specific formats*.

Chapter 6

Case study

This chapter contains the possible use of the implementation. We will compare all the modes and possibilities of the tool and summarize the result of each option. In the beginning, we will compare the results from the file to the existing solutions if possible (4). Then we will provide examples of the video quality monitoring in the *production network*.

6.1 Test results

In order to test the video quality measurement tool, we used three pcap capture files, each for a different type of a video transfer. The test files content is described in the following table:

	Size	**Type of a video service**	**Source**
prima_vod_new.pcap	33.3 MB	VoD	VoD captured while watching the Prima Play service
mpeg-ts.pcapng	674.5 MB	MPEG-TS	Captured stream of lectures on FIT
xlite_voip.pcapng	3.9 MB	VoIP	Captured VoIP call between two clients

Table 6.1: List of test pcap files

Then, we proceeded to measure the video quality from the capture files. Each testing capture file analysis output is described in a separate table.

prima_vod_new.pcap	**Port** 50351	**Port** 50361	**Port** 50405	**Port** 50375
Flow length [s]	2.3	5.22	32.6	3.35
Media bitrate [kB/s]	984.52	1102.34	392.42	147.72
Jitter [ms]	1.0064	0.4915	2.2229	2.8153
Delay [ms]	1.4697	1.3143	3.6942	9.7249
MDI	0.04988:0	0.01729:0	0.01569:0	0.01133:0
Codec	MPEG-4	MPEG-4	MPEG-4	MPEG-4

Table 6.2: Video on Demand test file

The first test file captured a Video on Demand packet flow from the PrimaPlay online television service. We can see that the video was being transmitted to different destination ports (from the same IP address) over time. The port *50361* shows the highest burst of packets to the client together with a very good video quality information. It might represent a video advertisement in the beginning of the video transfer from the server.

mpeg-ts.pcapng	**Flow** 1226:4468	**Flow** 4292:4444	**Flow** 4293:4446
Flow length [s]	211.879	211.862	211.85
Media bitrate [kB/s]	5372.53	5372.77	5372.79
Jitter [ms]	1.3886	1.3786	1.6532
Delay [ms]	1.1739	1.2228	1.2180
MDI	0.00099:0	0.001049:0	0.00092:0
Codec	H.262 (MPEG-2)	H.262 (MPEG-2)	H.262 (MPEG-2)

Table 6.3: Lecture streaming test file

The second test file captured a video stream over the MPEG-TS. All of the video flows had the same *PID*, which means that this capture might contain multiple lectures streamed at once. The possible higher media bitrate for each stream (which is generally higher for a MPEG Transport Stream) might be caused by the older coding algorithm type. The Jitter and Delay values are very low, which corresponds to a high stream quality. The capture file was probably captured on the same network it was created on and on a computer connected with a networking cable instead of the use of a IEEE 802.11 standard. There is also no packet loss detected, which can be see from the second field of a Media Delivery Index. Overall, the quality of this stream is very high.

The results were compared with the experimental results of a *Video Quality Monitor* tool (4.4), which produced hypothetical MOS results. These result values were in the range $< 8.75; 24 >$, which is an impossible range for a Mean Opinion Score, so we should not take them into account.

xlite_voip.pcapng	**Flow** 60902:53946	**Flow** 53946:60902
Flow length [s]	51.0986	51.2112
Media bitrate [kB/s]	17.4893	22.6861
Jitter [ms]	1.8707	3.6368
Delay [ms]	2.6452	5.8579
MDI	0.01132:0	2.1689:0
Codec	H.263	H.263

Table 6.4: VoIP call test file

The VoIP call test file contains two different video flows (the caller and the callee). The Media bitrate is very low compared to the MPEG-TS, but this is because of the substantiality of a VoIP call. We do not need the best video quality possible for a VoIP call, we just need to ensure that the audio and video is synchronized and that the video call is *coherent*. This is also possible with the use of a newer coding algorithm. There is a high value of a Time-stamped Delay-factor (in the Media Delivery Index field) in the second flow. This is due to the facts mentioned above in this paragraph. The main cause is their high difference between the maximum and minimum Delay captured (there were connectivity issues on the callee side).

The results were compared in a Wireshark program, where we can output statistics for a VoIP calls measuring Jitter and Codec.

6.2 Live capture testing

The live capture was tested for each mode.

- *Voice over IP*
  The `vidq` client running in the background of a video call. The output sometimes provided us with empty statistics to stdout, but the server did not accept such statistics and wrote the video related statistics correctly into relevant NetFlow files. We also captured the video call into a pcap file and checked the jitter and flows afterwards.
- *MPEG-TS*
  The MPEG Transport Stream was tested with a `tcpreplay` utility, which replayed the capture file and the `vidq` client produced statistics in live mode.
- *Video on Demand*
  Video on Demand has been tested on a live site, while watching different videos from a content provider. Every 5 minutes of a video watching, the `vidq` client sent the serialized video characteristics to the server, which compared the timestamps of present files and if the timestamp was in range, the server tried to find a matching record and update it. The Video on Demand services tested for the purpose of our work were as follows:
  1. iVysílání České televize a.s. http://www.ceskatelevize.cz/ivysilani/
  2. Prima Play FTV Prima s.r.o. http://play.iprima.cz/

In the Video on Demand case, I focused on the video quality settings comparison. The higher the quality selected (max 720p), the higher was the Media bitrate, while the Jitter, Delay and Media Delivery Index remained the same.

6.3 NetFlow record update testing

The last part tested in order to ensure the functionality of the tool was the NetFlow update module, which is made of a server updating the flows and the collector, which produces the NetFlow files. Because of the fact that I was testing the product in a production network with no access to a router able to export NetFlow packets to the collector, I had to install a network NetFlow `nProbe` by *ntop* company (5.1.5), which generated the NetFlow packets and sent them to the `nfcapd` (5.1.3) collector. This probe was a solution for the live network packet capture. In order to generate NetFlow packets from the pcap capture files, I used the `softflowd` tool (5.1.4). The NetFlow packets had to support the *username* field, because that is the field containing the video quality characteristics.

The server updating the NetFlow records informs us about the file which has been written in and afterwards we can check the content of the file as follows:

```
nfdump -r <nfcapd.timestamp> -o "fmt:  %sa:%sp %da:%dp %pkt %uname"
```

Note that the `nfdump` has to be compiled with a *NSEL/ASA support* in order to correctly show the *username* field.

Chapter 7

Conclusion

The process of a video quality assessment has to be very fast in order to be effective. That is why we have to filter out packets we are not interested in (not containing any video data) and apply a deep packet inspection on the video packets. Our video quality assessment uses a no-reference method (2), which means, that we do not have the reference video available. It is an objective method, which produces a set of results related to the video quality impairment. The video metrics we inspect for a video data flow are Media Delivery Index, consisting of Media Loss Rate and Time-stamped Delay Factor, One-way delay, Cumulative Inter-Arrival Jitter and Media Bitrate (3.2.2). These metrics can be combined with the video quality assessment from the codec information, because the quality detected from those metrics is dependant on the video coding algorithm. The implemented video quality measurement tool consists of a client side and server side application. Server side is a `nfcapd` tool (5.1.3) running as a collector together with the implemented `vidq_updater`, which updates the detected video flows in the relevant NetFlow records. Client side is a program called `vidq`, which can be used for an online monitoring or for an offline analysis of the flows captured in the pcap file. Result of the measurement and analysis from the client `vidq` program is a serialized string (5.2.1) exported to the server, where the `vidq_updater` is running.

We also looked into other works, which might be related to our project. We selected four of the closest candidates, but these tools usually use the full-reference metrics and need to input the original non-compressed file to compare the video quality. Another candidate was producing incorrect values for the common metrics (MOS) and the scale was different. The *ProLab* solution was mentioned, but not tested because of the commercial licence and hardware implementation, which could not be compared to our work.

The video quality measurement system was tested in the production network. In the beginning, we compared the results with the tools related to ours (if the comparison was possible) and then we tested the tools for each type of service mentioned in 2.8. The final product updates the NetFlow records on a collector via the server, which accepts the video quality statistics in a serialized string from the client. It can either update the existing flows from the captured pcap files or update the relevant NetFlow records captured in a specific time horizon from the online video quality monitoring.

Bibliography

[1] ALMES, G.; KALIDINDI, S.; ZEKAUSKAS, M. J.: *RFC 2680*. 1999.
Retrieved from: https://tools.ietf.org/html/rfc2680

[2] BORMANN, C.; Linda CLINE, G. D.; GARDOS, T.; et al.: *RFC 2429*. 1998.
Retrieved from: https://tools.ietf.org/html/rfc2429

[3] BRYANT, S.; LEINEN, S.; DIETZ, T.: *RFC 7011*. 2013.
Retrieved from: https://tools.ietf.org/html/rfc7011

[4] CLAISE, B.: *Cisco Systems NetFlow Services Export Version 9*. 2004. IETF RFC 3954.

[5] DARPA: *RFC 793*. 1981.
Retrieved from: https://tools.ietf.org/html/rfc793

[6] DEERING, S. E.; HINDEN, R. M.: *RFC 2460*. 1998.
Retrieved from: https://tools.ietf.org/html/rfc2460

[7] Dictionary.com: *Objective: Definition*. [Online; navštíveno 11.1.2017].
Retrieved from: http://www.dictionary.com/browse/objective

[8] E. L. LEHMANN, G. C.: *Theory of Point Estimation*. Springer. second edition. ISBN 0-387-98502-6.

[9] EASTLAKE, D.: *RFC 5342*. 2008.
Retrieved from: https://tools.ietf.org/html/rfc5342

[10] EVEN, R.: *RFC 4587*. 2006.
Retrieved from: https://tools.ietf.org/html/rfc4587

[11] EXFO: Delay Factor (DF).
Retrieved from: http://www.exfo.com/glossary/delay-factor-df

[12] FAIRHURST, G.; COLLINI-NOCKER, B.: *RFC 4326*. 2003.
Retrieved from: https://tools.ietf.org/html/rfc4326

[13] FIELDING, R. T.; GETTYS, J.; MOGUL, J. C.; et al.: *RFC 2616*. 1999.
Retrieved from: https://tools.ietf.org/html/rfc2616

[14] FTV Prima, s. s.: Prima PLAY.
Retrieved from: http://play.iprima.cz/

[15] HANDLEY, M.; JACOBSON, V.; PERKINS, C.: *RFC 4566*. 2006.
Retrieved from: https://tools.ietf.org/html/rfc4566

[16] ICANN: Available Pool of Unallocated IPv4 Internet Addresses Now Completely Emptied.
Retrieved from: https://www.icann.org/en/system/files/press-materials/release-03feb11-en.pdf

[17] IEEE: *EtherType*.
Retrieved from: http://standards-oui.ieee.org/ethertype/eth.txt

[18] IEEE: *802.3-2015*. March 2016. doi:10.1109/IEEESTD.2016.7428776.
Retrieved from: http://ieeexplore.ieee.org/document/7428776/

[19] Inc., G.: Youtube.
Retrieved from: https://www.youtube.com

[20] info, V.: SIP method ack.
Retrieved from: http://www.voip-info.org/wiki/view/SIP+method+ack

[21] Institute, I. S.: *RFC 791*. 1981.
Retrieved from: https://tools.ietf.org/html/rfc791

[22] ISO/IEC: *23009-1:2014, Dynamic adaptive streaming over HTTP (DASH) – Part 1: Media presentation description and segment formats*.
Retrieved from: http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=65274

[23] ITU-T: *J.246 : Perceptual visual quality measurement techniques for multimedia services over digital cable television networks in the presence of a reduced bandwidth reference*.
Retrieved from: http://www.itu.int/rec/T-REC-J.246/en

[24] ITU-T: *J.247 : Objective perceptual multimedia video quality measurement in the presence of a full reference*.
Retrieved from: http://www.itu.int/rec/T-REC-J.247/en

[25] ITU-T: *P.10: Vocabulary for performance and quality of service*.
Retrieved from: https://www.itu.int/rec/T-REC-P.910-200804-I/en

[26] ITU-T: *P.910: Subjective video quality assessment methods for multimedia applications*.
Retrieved from: https://www.itu.int/rec/T-REC-P.10-200607-I/en

[27] James, W.; James, C.: *RFC 4445*. 2006.
Retrieved from: https://tools.ietf.org/html/rfc4445

[28] LIPSCOMBE, A.; PROWSE, C.; KERNEN, T.; et al.: A Proposed Time-Stamped Delay Factor (TS-DF) algorithm for measuring Network Jitter on RTP Streams. Geneva.
Retrieved from: https://tech.ebu.ch/docs/tech/tech3337.pdf

[29] MATOUSEK, P.; KMET, M.; BASEL, M.: On-line Monitoring of VoIP quality using IPFIX. ISSN 13361376. doi:0.15598/aeee.v4i4.1203. brno University of Technology.

[30] MATOUŠEK, P.: *Síťové aplikace a jejich architektura*. VUTIUM. 2014. ISBN 978-80-214-3766-1. Chapter 1: ARCHITEKTURA SÍTÍ, ADRESOVÁNÍ, KONFIGURACE TCP/IP.

[31] MEIER, B.: TCP Relative Sequence Numbers and the TCP Window Scaling.
Retrieved from: https://wiki.wireshark.org/TCP_Relative_Sequence_Numbers

[32] Mindrot: Softflowd.
Retrieved from: http://www.mindrot.org/projects/softflowd/

[33] Monitor, V. Q.: AccepTV.
Retrieved from: http://www.acceptv.com/en/products_vqm.php

[34] MONTPETIT, M. J.; FAIRHURST, G.; CLAUSEN, H. D.; et al.: *RFC 4259*. 2005.
Retrieved from: https://tools.ietf.org/html/rfc4259

[35] ntop: nProbe^{TM} An Extensible NetFlow v5/v9/IPFIX Probe for IPv4/v6.
Retrieved from: http://www.ntop.org/products/netflow/nprobe/

[36] OPTICOM: PEVQ – the Standard for Perceptual Evaluation of Video Quality.
Retrieved from: http://www.pevq.com/pevq.html

[37] PATTERSON, M.: What is IPFIX vs. NetFlow v9?
Retrieved from:
https://www.plixer.com/blog/netflow/what-is-ipfix-vs-netflow-v9/

[38] PORTER, D. T.: The Perils of Deep Packet Inspection.
Retrieved from:
https://www.symantec.com/connect/articles/perils-deep-packet-inspection

[39] POSTEL, J.: *RFC 768*. 1980.
Retrieved from: https://tools.ietf.org/html/rfc768

[40] SALOMON, D.: *Data Compression: The Complete Reference*. Springer. fourth edition. ISBN 978-1846286025.

[41] SCHULZRINNE, H.; CAMARILLO, G.; JOHNSTON, A.; et al.: *RFC 3261*. 2002.
Retrieved from: https://tools.ietf.org/html/rfc3261

[42] SCHULZRINNE, H.; CASNER, S. L.: *RFC 3551*. 2003.
Retrieved from: https://tools.ietf.org/html/rfc3551

[43] SCHULZRINNE, H.; CASNER, S. L.; FREDERICK, R.; et al.: *RFC 3550*. 2003.
Retrieved from: https://tools.ietf.org/html/rfc3550

[44] SCHULZRINNE, H.; RAO, A.; LANPHIER, R.: *RFC 2326*. 1998.
Retrieved from: https://tools.ietf.org/html/rfc2326

[45] SPIRENT: ProLab - Audio and Video Quality Testing Solution.
Retrieved from:
https://www.spirent.com/Products/ProLab/ProLab_Video-Quality

[46] STRETCH, J.: Understanding TCP Sequence and Acknowledgment Numbers.
Retrieved from: http://packetlife.net/blog/2010/jun/7/understanding-tcp-sequence-acknowledgment-numbers/

[47] SUN, L.; MKWAWA, I.-H.; JAMMEH, E.; et al.: *Guide to voice and video over IP: for fixed and mobile networks*. Springer-Verlag. ISBN 978-1-4471-4905-7.

[48] Technologies, A.: IPTV QoE: Understanding and interpreting MDI values.
Retrieved from: http://cp.literature.agilent.com/litweb/pdf/5989-5088EN.pdf

[49] Česká televize: iVysílání.
Retrieved from: http://www.ceskatelevize.cz/ivysilani/

[50] VATOLIN, D. D.; MOSKVIN, A.; PETROV, O.; et al.: MSU Video Quality Measurement Tool.
Retrieved from:
http://compression.ru/video/quality_measure/video_measurement_tool.html

[51] WANG, Z.; BOVIK, A.; SHEIKH, H.; et al.: *Image quality assessment: from error visibility to structural similarity*. IEEE. doi:10.1109/TIP.2003.819861.

[52] WEBSTER, A.; PINSON, M.: Video Quality Experts Group (VQEG).
Retrieved from: https://www.its.bldrdoc.gov/vqeg/vqeg-home.aspx

[53] ZHU, C. C.: *RFC 2190*. 1997.
Retrieved from: https://tools.ietf.org/html/rfc2190

Appendices

Video quality measurement setup

Client

To install a client from the archive, extract the files and compile the source files with the `make` command.

If the makefile fails with the `pcap.h fatal error`, please install the libpcap library with the following command:

`sudo apt-get install libpcap-dev` on the Debian packages and
`sudo yum install libpcap-devel` on the Fedora/CentOS packages

Client for video quality measurement on local machine can be run as follows:

Off-line measurement from a pcap file

```
./vidq -f <filename.pcap> [-a <hostname:port>] [-v] [-t]
```

- `-f <filename>` Mandatory argument. Specifies the pcap file for a video quality analysis
- `-a <hostname:port>` Optional argument. Specifies the server address and port for data export. If this option isn't present, `VIDQ` prints statistics to the `stdout`
- `-v` Optional argument. Verbose mode, prints more detailed output to the `stdout` (e.g.: SIP INVITE request detection, etc.)
- `-t` Optional argument. No timestamp, after sending statistics to server, server iterates through all the files

On-line measurement on a live network

```
./vidq -l [-a <hostname:port>] [-v] [-t]
```

- `-l` Mandatory argument. Starts a live capture. You have to be the root to run the program with this option
- `-a <hostname:port>` Optional argument. Specifies the server address and port for data export. If this option isn't present, `VIDQ` prints statistics to the `stdout`
- `-v` Optional argument. Verbose mode, prints more detailed output to the `stdout` (e.g.: SIP INVITE request detection, etc.)
- `-t` Optional argument. No timestamp, after sending statistics to server, server iterates through all the files

Print help and end

```
./vidq -h
```

Server

Client for video quality measurement on local machine can be run as follows:

Starting a server

```
./vidq_server <port-number> -d <dir>
```

- `<port-number>` Mandatory argument. Specifies the port number which the server is listening on
- `-d <dir>` Mandatory argument. Specifies the directory with NetFlow record files

Make sure the collector is running on the same node as the `VIDQ_UPDATER` program (or has access to the collector output NetFlow record files).

Installation

Client

For an installation of the client side video quality measurement tool `vidq`, compile the source files with a `make` command. Existing binary and objective files can be removed with the `make clean` command.

Server

For a first installation of a server NetFlow updater, run the following command:

```
sh configure.sh
```

This script will prepare the environment for you. If another compilation is needed (for example after the `make clean` command), compile the source files with the `make` command.

If there is any problem with linking the `libnf` shared library, try to run the `configure` script again and add the following command afterwards:

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:./libnf-1.25/usr_bin/lib/
```

Usage examples

1. *Analyse pcap file and output video quality statistics to stdout*

   ```
   ./vidq -f test.pcap
   ```

2. *Analyse live capture and output video quality statistics to stdout*

   ```
   sudo ./vidq -l
   ```

3. *Analyse live capture and output video quality statistics to stdout with a verbose mode*

   ```
   sudo ./vidq -l -v
   ```

4. *Generate NetFlow records from a pcap file and update these records with video quality statistics from a pcap file sent to a server*

Exporter: `softflowd -r test.pcap -n 127.0.0.1:9995 -d`
Collector: `nfcapd -b 127.0.0.1 -p 9995 -l netflow_records_dir/`
Server-updater: `./vidq-server 12345 -d netflow_records_dir/`
Client: `./vidq -f test.pcap -a 127.0.0.1:12345`

5. *Update existing relevant NetFlow records with video quality statistics from a pcap file with any timestamp (cycles thorugh all files)*

   Server-updater: `./vidq-server 12345 -d netflow_records_dir/`
   Client: `./vidq -f test.pcap -a 127.0.0.1:12345 -t`

6. *Use NetFlow probe to export NetFlow records and update those records with video quality statistics from a live capture sent to a server*

   Probe: `sudo nprobe -collector-port any -collector 127.0.0.1:9995 -V 10` or Exporter `sudo softflowd -i wlan0 -n 127.0.0.1:9995 -v 9 -d` Collector: `nfcapd -b 127.0.0.1 -p 9995 -l netflow_records_dir/`
   Server-updater: `./vidq-server 12345 -d netflow_records_dir/`
   Client: `sudo ./vidq -l -a 127.0.0.1:12345`